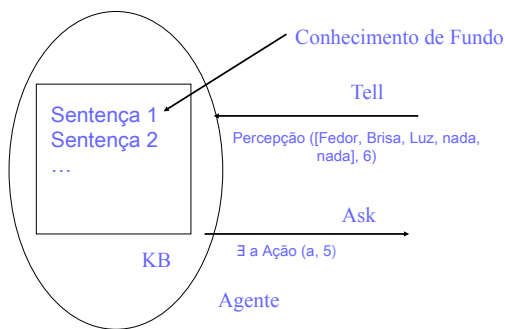
PCS 2428 / PCS 2059
Inteligência Artificial

Prof. Dr. Jaime Simão Sichman
Prof. Dra. Anna Helena Reali Costa

Cálculo de Situações

Agente Baseado em Conhecimento

função Agente-Baseado-Conhecimento(*percepção*)
  retorna uma *ação*
  estático: base de conhecimento BC, contador t =0
  Tell(*BC*, Sentenças-Percepções(*percepção,t*))
  ação ← Ask(*BC*, Pergunte-Ação(*t*))
  Tell(*BC*, Sentença-Ação(*ação,t*))
  t ← *t* + 1
  retorna *ação*

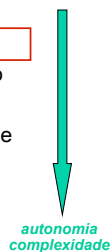
Agente Baseado em Lógica de Predicados

Um Agente LPO para o Mundo do Wumpus

- Interface entre o agente e o ambiente:
  - sentenças que representam percepções, incluindo seus valores e o tempo (passo) em que elas ocorreram:
    Percepção ([Fedor, Brisa, Luz, Nada, Nada], 6)
- Ações do agente:
  - constantes que representem as diversas ações possíveis:

GirarDireita	GirarEsquerda
Pegar	Soltar
Avançar	Sair
Atirar	

Arquiteturas de Agentes

- Agente tabela
- Agente reativo
- Agente baseado em modelo
- Agente baseado em metas
- Agente baseado em utilidade
- Agente aprendiz

autonomia
complexidade

Agente Reativo Baseado em LPO

- Possui regras ligando as seqüências de percepções às ações
  $\forall$ f,b,c,g,t Percepção ([f,b,Luz,c,g], t) → Ação (Pegar, t)
- Poderia dividir tais regras em duas classes:
  - Regras de interpretação da percepção
    $\forall$ b,l,c,g,t Percepção ([Fedor,b,l,c,g], t) → Fedor (t)
    $\forall$ f,l,c,g,t Percepção ([f,Brisa,l,c,g], t) → Brisa (t)
    $\forall$ f,b,c,g,t Percepção ([f,b,Luz,c,g], t) → Junto-do-Ouro (t)
  - Regras de ação
    $\forall$ t Junto-do-Ouro (t) → Ação (Pegar, t)

Limitações do Agente Reativo

- Um agente ótimo deveria:
  - recuperar o ouro ou
  - determinar que é muito perigoso pegar o ouro e
  - em qualquer dos casos acima, voltar para (1,1) e sair da caverna.
- Um agente reativo nunca sabe quando sair,
  - estar com o ouro e estar na caverna (1,1) não fazem parte da sua percepção (a percepção só indica que reluz quando o ouro está na caverna; agente não sabe onde se encontra).
  - esses agentes podem entrar em laços infinitos.
- Para ter essas informações, o agente precisa guardar uma representação do mundo.

Agentes LPO com Modelo do Mundo

- Um agente terá um comportamento ótimo se:
  - todas as percepções são gravadas na BC, e
  - existem regras para lidar com percepções passadas e presentes.

  Porém, escrever essas regras dá trabalho e é ineficiente!
- Solução:
  - modelo interno do mundo = sentenças sobre o mundo atual, em vez de percepções passadas
    - "às 4h30 pegou o ouro" ===> "está com o ouro"
  - as sentenças serão atualizadas quando:
    - receber novas percepções e realizar ações
    - ex. chaves no bolso, pegou o ouro,..

Representando Mudanças no Mundo

Como representar as mudanças realmente?
- O agente foi de [1,1] para [1,2]

1. Apagar" da BC sentenças que já não são verdade
   - **ruim**: perdemos o conhecimento sobre o passado, o que impossibilita previsões de diferentes futuros.
2. O agente pode buscar no espaço de estados passados e (possíveis) futuros, onde cada estado é representado por uma BC diferente:
   - **ruim**: pode explorar situações hipotéticas, porém não pode raciocinar sobre mais de uma situação ao mesmo tempo.
     - ex. "existiam buracos em (1,2) e (3,2)?"

Solução: Cálculo Situacional
- uma maneira de escrever mudanças no tempo em LPO.
- representação de diferentes **situações** na mesma BC.

Cálculo de Situações

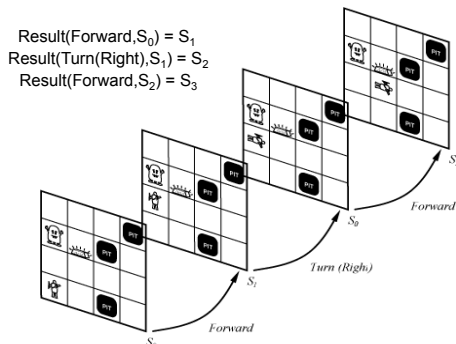
- O mundo consiste em uma seqüência de situações
  - situação N ===**ação**===> situação N+1
- Predicados que **mudam** com o tempo têm um argumento de **situação** adicional

  Em(Agente,[1,1],S0) ∧ Em(Agente,[1,2],S1)
- Predicados que denotam propriedades que **não mudam** com o tempo não usam argumentos de situação

  Parede(0,1) ∧ Parede(1,0)
- Para representar as mudanças no mundo, usa-se uma função Resultado

  Resultado (ação,situação N) = situação N+1

Cálculo de Situações

Result(Forward,S_0) = S_1
Result(Turn(Right),S_1) = S_2
Result(Forward,S_2) = S_3

Representando Efeito das Ações

- As **ações** são descritas pelos seus efeitos
  - Especificam-se as propriedades da situação resultante da realização da ação
- Exemplo:
  - Caso o agente esteja em uma situação onde ele esteja numa posição onde haja o ouro e se ele escolher realizar a ação Pegar, na situação seguinte ele estará segurando o ouro

    Portável(Ouro)

    ∀ s Junto-do-Ouro (s) → Presente (Ouro, s)

    ∀ x, s Presente (x, s) ∧ Portável (x) → Segurando (x, Resultado (Pegar, s))
- Usam-se **axiomas de efeito** e **axiomas de quadro**.

Axiomas de Efeito

- **Axiomas de efeito**: descrevem as propriedades do mundo que mudam após uma ação
- Exemplo:
  - O agente estará segurando algo se ele acabou de pegá-lo
    $\forall$ x, s Presente(x, s) $\wedge$ Portável(x) $\rightarrow$ Segurando(x, Resultado(Pegar,s))
  - O agente não estará segurando nada depois de realizar uma ação de Soltar
    $\forall$ x, s $\neg$ Segurando(x, Resultado(Soltar,s))

Axiomas de Quadro

- **Axiomas de quadro**: descrevem as propriedades do mundo que **não** mudam após uma ação
- Exemplo:
  - O agente continuará segurando algo se ele não realizou a ação de Soltar
    $\forall$ a, x, s Segurando (x, s) $\wedge$ (a $\neq$ Soltar) $\rightarrow$ Segurando(x, Resultado(a, s))
  - O agente não estará segurando nada depois de realizar qualquer ação de distinta de Pegar
    $\forall$ a, x, s $\neg$ Segurando (x, s) $\wedge$
    ((a $\neq$ Pegar) $\vee$ $\neg$(Presente (x, s) $\wedge$ Portável (x)) $\rightarrow$
    $\neg$ Segurando(x, Resultado(a, s))

Axiomas Estado-Sucessor

- **Axioma estado-sucessor:** combinação entre os axiomas de efeito e de quadro
  uma coisa é verdade depois $\leftrightarrow$
  [uma ação acabou de torná-la verdade
  $\vee$
  ela já era verdade e nenhuma ação a tornou falsa]
- Exemplo:
  $\forall$ a, x, s Segurando (x, Resultado(a,s)) $\leftrightarrow$
  [(a = Pegar $\wedge$ Presente (x, s) $\wedge$ Portável (x))
  $\vee$ (Segurando (x, s) $\wedge$ (a $\neq$ Soltar)]
- É necessário escrever um axioma estado-sucessor para cada predicado que pode mudar seu valor no tempo.

Problema do Quadro

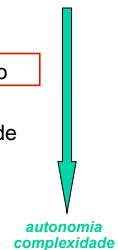
- Chamado de "Frame Problem"
- Problema de **representação** dos axiomas de frame:
  - proliferação de axiomas de frames
  - tema de calorosos debates durante anos
  - solução: uso de axiomas estado-sucessor
- Problema de **inferência** dos axiomas de frame:
  - excesso de inferências (para atualizar todo o mundo)
  - solução: usar sistemas de planejamento (só atualizam as partes do estado estritamente necessárias)

Outros Problemas Relacionados

- Problema de Qualificação
  - Chamado de Qualification Problem
  - dificuldade em enumerar todas as pré-condições de **sucesso** de uma ação
  - Exemplo: O agente estará segurando o ouro se ele acabou de pegá-lo **e o ouro não escorregar e o ouro não estiver grudado na caverna e ....**
- Problema da Ramificação
  - Chamado de Ramification Problem
  - dificuldade em enumerar todos os efeitos **implícitos** de uma ação
  - Exemplo: Se o ouro estiver carregado de poeira, quando o agente o pegar, **ele também estará pegando a poeira**

Arquiteturas de Agentes

- Agente tabela
- Agente reativo
- Agente baseado em modelo
- Agente baseado em metas
- Agente baseado em utilidade
- Agente aprendiz

autonomia
complexidade

Agente LPO - Guardando Localizações

- O agente precisa lembrar por onde andou e o que viu
  - para deduzir onde estão os buracos e o Wumpus, e
  - para garantir uma exploração completa das cavernas
- O agente precisa saber:
  - **localização inicial** = predicado que indica onde está
    Em (Agente, [1,1], S0)
  - **orientação**: função que indica sua direção (em graus)
    Orientação (Agente,S0) = 0

Agente LPO - Guardando Localizações

- O agente precisa ainda saber:
  - **próximas localizações possíveis:** função de locais e orientações
    ∀ x,y PróximaLocalização ([x,y], 0) = [x+1, y]
    ∀ x,y PróximaLocalização ([x,y], 90) = [x, y+1]
    ∀ x,y PróximaLocalização ([x,y], 180) = [x-1, y]
    ∀ x,y PróximaLocalização ([x,y], 270) = [x, y-1]

Agente LPO - Guardando localizações

- A partir desses axiomas, pode-se deduzir qual célula está em frente ao agente numa localização "l":
  ∀ ag, l, s Em (ag, l, s) → LocalizaçãoEmFrente (ag,s) = PróximaLocalização (l, Orientação (ag,s))
- Pode-se também definir a noção de adjacência:
  ∀ l_1,l_2 Adjacente (l_1,l_2) ↔ ∃ d l_1 = PróximaLocalização (l_2,d)
- Pode-se indicar detalhes geográficos do mapa:
  ∀ x,y Parede([x,y]) ↔ (x =0 ∨ x =5 ∨ y =0 ∨ y =5)

Axioma Estado-Sucessor para Localização

- Resultado das ações sobre a localização do agente:
  - Axioma Estado-Sucessor: **avançar** é a única ação que muda a localização do agente (a menos que haja uma parede)

∀ a, l, ag, s Em (ag, l, Resultado (a,s)) ↔
[(a = Avançar ∧ l =LocalizaçãoEmFrente (ag, s) ∧ ¬parede (l))
∨
(Em (ag, l, s) ∧ a ≠ Avançar)]

Axioma Estado-Sucessor para Direção

- Resultado das ações sobre a orientação do agente:
  - Axioma Estado-Sucessor: **girar** é a única ação que muda a direção do agente

∀ a, d, ag, s Orientação (ag, Resultado(a,s)) = d ↔
[(a=GirarDireita ∧ d=Mod(Orientação(ag,s)-90, 360) ∨
(a=GirarEsquerda ∧ d= Mod(Orientação(ag,s) + 90, 360)) ∨
(Orientação(ag,s)=d ∧ ¬(a=GirarDireita ∨ a=GirarEsquerda))]

Deduzindo Propriedades do Mundo

- Agora que o agente sabe onde está, ele pode associar propriedades aos locais:
  ∀ l,s Em(Agente, l, s) ∧ Brisa(s) → Ventilado(l)
  ∀ l,s Em(Agente, l, s) ∧ Fedor(s) → Fedorento(l)
- Sabendo isto o agente pode deduzir:
  - onde estão os buracos e o Wumpus, e
  - quais são as cavernas seguras (predicado *OK*).
- Os predicados *Ventilado* e *Fedorento* não necessitam do argumento de situação

Tipos de Regras

- Regras diacrônicas
  - do grego: "através do tempo"
  - descrevem como o mundo evolui (muda ou não) com o **tempo**

$\forall$ x,s Presente(x,s) $\wedge$ Portável(x) $\Rightarrow$ Segurando(x, Resultado (Pegar, s))

- Regras Síncronas:
  - relacionam propriedades na mesma situação (tempo).
  - existem dois tipos principais de regras síncronas:
    - **Regras Causais**: deduzem efeitos de causas
      $\forall$ x,y Buraco(x) $\wedge$ Adjacente(x,y) $\Rightarrow$ Ventilado(y)
    - **Regras de Diagnóstico**: *deduzem causas de efeitos*
      $\forall$ y Ventilado(y) $\Rightarrow$ $\exists$x Buraco(x) $\wedge$ Adjacente(x,y)
- Definição para o predicado Ventilado:
  $\forall$ y Ventilado(y) $\Leftrightarrow$ ($\exists$x Buraco(x) $\wedge$ Adjacente(x,y))

Modularidade das Regras

- As regras que definimos até agora não são **modulares**:
  - Para torná-las mais modulares, separamos fatos sobre ações de fatos sobre objetivos:
    - Ações descrevem como alcançar resultados.
    - Objetivos descrevem a adequação (desirability) de estados resultantes, não importando como foram alcançados.
  - Assim, o agente pode ser "reprogramado" mudando-se o seu **objetivo**.
- Descreve-se o grau de adequação das regras para que que a máquina de inferência escolha a ação mais adequada.

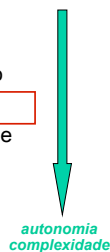
Grau de Adequação das Regras

- Uma possível escala, em grau decrescente de adequação:
  - ações podem ser: *ótimas, boas, médias, arriscadas* e *mortais*.
  - *O agente escolhe a mais adequada:*
    $\forall$ *a, s Ótima (a, s)* $\rightarrow$ *Ação (a, s)*
    $\forall$ *a, s Boa (a, s)* $\wedge$ ($\neg$ $\exists$ *b Ótima (b, s)*) $\rightarrow$ *Ação (a, s)*
    $\forall$ *a, s Média (a, s)* $\wedge$ ($\neg$ $\exists$ *b* (*Ótima (b, s)* $\vee$ *Boa (b, s)*)) $\rightarrow$ *Ação (a, s)*
    $\forall$ *a, s Arriscada (a, s)* $\wedge$ ($\neg$ $\exists$ *b* (*Ótima (b, s)* $\vee$ *Boa (b, s)* $\vee$ *Média (a, s)*)) $\rightarrow$ *Ação (a, s)*
- Essas regras são gerais, podem ser usadas em situações diferentes:
  - uma ação **arriscada** na situação S0, onde o Wumpus está vivo, pode ser **ótima** na situação S2, quando o Wumpus já está morto.

Sistema de Ação-Valor

- Sistema de ação-valor: **é um sistema baseado em regras de adequação**
  - Não se refere ao que a ação **faz**, mas a quão **desejável** ela é.
- Prioridades do agente até encontrar o ouro:
  - **ações ótimas:** pegar o ouro quando ele é encontrado, e sair das cavernas.
  - **ações boas:** mover-se para uma caverna que está OK e ainda não foi visitada.
  - **ações médias:** mover-se para uma caverna que está OK e já foi visitada.
  - **ações arriscadas**:mover-se para uma caverna que não se sabe com certeza que não é mortal, mas também não é OK
  - **ações mortais**: mover-se para cavernas que sabidamente contêm buracos ou o Wumpus vivo.

Arquiteturas de Agentes

- Agente tabela
- Agente reativo
- Agente baseado em modelo
- Agente baseado em metas
- Agente baseado em utilidade
- Agente aprendiz

Em Direção a Agentes Baseados em Metas

- O conjunto de ações-valores é suficiente para prescrever uma boa estratégia de exploração inteligente das cavernas.
  - quando houver uma seqüência segura de ações , ele acha o ouro
  - Porém... isso é tudo o que um agente baseado em LPO pode fazer.
- Depois de encontrar o ouro, a estratégia deve mudar...
  - novo objetivo: estar na caverna (1,1) e sair.
    $\forall$ s Segurando(Ouro,s) $\rightarrow$ LocalObjetivo ([1,1],s)
- A presença de um objetivo explícito permite que o agente encontre uma seqüência de ações que alcançam esse objetivo.

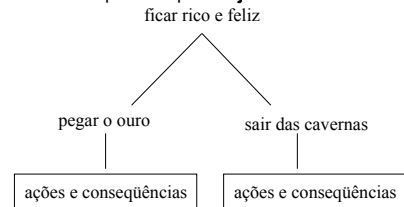
Como encontrar seqüências de ações?

(1) Inferência:
- **idéia**: escrever axiomas que perguntam à BC uma seqüência de ações que com certeza alcança o objetivo.
- **Porém:** para um mundo mais complexo isto se torna muito difícil.
- como distinguir entre boas soluções e soluções mais dispendiosas (onde o agente anda "à toa" pelas cavernas)?

(2) Busca:
- Usar Busca pela Melhor Escolha (best-first) para encontrar um caminho até o objetivo.
- Nem sempre é fácil traduzir conhecimento em um conjunto de operadores, e representar o problema (ambiente) em estados para poder aplicar o algoritmo.

Agentes Baseados em Objetivos

(3) Planejamento:
- envolve o uso de um sistema de raciocínio dedicado, projetado para raciocinar sobre ações e conseqüências para **objetivos** diferentes.

ficar rico e feliz
pegar o ouro
sair das cavernas
ações e conseqüências
ações e conseqüências

Referências Bibliográficas

- S. Russel and P. Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River, USA. 2nd. Edition, 2003. Chapter 8 and 9.
- G. Bittencourt. Inteligência Artificial: Ferramentas e Teorias. Editora da UFSC, Florianópolis. 2a. Edição, 2001. Cap. 3.